

# The embedfile package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2007/10/28 v2.0

## Abstract

This package embeds files to a PDF document. Currently the only supported driver is pdfTeX >= 1.30 in PDF mode.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Introduction	2
1.1.1	Future development	2
1.2	User interface	2
1.3	Collection support (PDF 1.7)	3
1.4	Examples	4
1.4.1	plain-T <sub>E</sub> X	4
1.4.2	Collection example	5
1.5	Package dtx-attach	6
<b>2</b>	<b>Implementation</b>	<b>6</b>
2.1	Reload check and package identification	7
2.2	Catcodes	7
2.3	Tools	8
2.4	Check for recent pdfTeX in PDF mode	8
2.5	Strings	9
2.6	Switches	10
2.7	Key value definitions	10
2.8	Embed the file	16
<b>3</b>	<b>Test</b>	<b>20</b>
3.1	Catcode checks for loading	20
3.2	Simple test	21
<b>4</b>	<b>Installation</b>	<b>21</b>
4.1	Download	21
4.2	Bundle installation	22
4.3	Package installation	22
4.4	Refresh file name databases	22
4.5	Some details for the interested	22
<b>5</b>	<b>References</b>	<b>23</b>
<b>6</b>	<b>History</b>	<b>23</b>
	[2006/08/16 v1.0]	23
	[2007/04/11 v1.1]	23
	[2007/09/09 v1.2]	23
	[2007/10/28 v2.0]	23

# 1 Documentation

## 1.1 Introduction

The PDF format ([3]) allows the inclusion of files inside the PDF document. The included files can be bound to an annotation on a page. Or they can be recorded in a sorted list of embedded files. The packages `attachfile` or `attachfile2` follow the first approach, this package uses the latter method.

### 1.1.1 Future development

My dream is a large package that merges the features of all these packages mentioned before:

- Files can be attached to a page.
- Files can be attached to the document.
- An easy user interface for simple, common tasks and beginners.
- An interface for the advanced users that want to setup every detail.
- Support of many drivers (`pdftex`, `dvips`, `dvipdfm`, ...).
- ...

However, I have not managed to take the time for this project. Instead:

- First I experimented with package `attachfile`, adding driver support, fixing bugs, .... The result is currently named as `attachfile2`. It uses an external script to get file properties (size, date, checksum, ...).
- In order to avoid an external program for getting basic file properties I provided a patch “EscapeAndOther” for pdfTeX that was accepted for version 1.30.
- This package closes a gap left by the packages for attaching files and allows the embedding of files to the document. Also it makes use of the new primitives of pdfTeX.

## 1.2 User interface

This package `embedfile` can be used with both L<sup>A</sup>T<sub>E</sub>X and plain-T<sub>E</sub>X. See [subsection 1.4.1](#) that explains the use with plain-T<sub>E</sub>X by an example. In L<sup>A</sup>T<sub>E</sub>X the package is loaded as usually. There are no options.

```
\usepackage{embedfile}
```

```
\embedfile [options] {file}
```

The macro `\embedfile` includes file *file* and attaches it to the PDF document. At the end of the document the sorted list of embedded files are written. Thus you can safely use `\embedfile` before `\end{document}`. Embedding files using `\AtEndDocument` will only work, if `\AtEndDocument` is called before loading the package `embedfile`.

The *options* are give as key value pairs. The following keys are supported:

**filespec** This allows to override the file name that appears in the PDF file. If you are using other than simple file names (8bit, path separators, ...), look into the PDF specification ([3]). There are rules how these file names must be written/encoded.

**filesystem** This sets the entry /FS in the file specification dictionary, see PDF specification ([3]). Example: `filesystem=URL`.

**mimetype** This sets the mime type ([4]) of the file, see [subsubsection 1.4.1](#) for examples and [5] for a list of officially registered types.

**desc** The description for the file.

**stringmethod** The package must convert the values of the keys `filespec` and `desc` into a PDF string. If `hyperref` is found, then its `\pdfstringdef` will be used, otherwise pdfTeX’s `\pdfescapestring` is used. Value `psd` forces the use of `\pdfstringdef`, value `escape` the use of `\pdfescapestring`.

**<key>.value** Sets the value of a collection item property, see [section 1.3](#).

**<key>.prefix** Sets the prefix of a collection item property, see [section 1.3](#).

`\embedfilefinish`

The list of all embedded files must be added as data structure in the PDF file. In case of L<sup>A</sup>T<sub>E</sub>X this is automatically done. The package uses `\AtEndDocument`. Then the list of all files should be known. However, plain-T<sub>E</sub>X does not know about `\AtEndDocument`. Thus the user must call `\embedfilefinish` at the end of the document after the last file is embedded.

`\embedfilesetup {<options>}`

Options for `\embedfile` and collection support can be set in `\embedfilesetup`.

### 1.3 Collection support (PDF 1.7)

Since PDF 1.7 the embedded files can form a *collection* (sometimes referred as *package*), the main document is called *cover sheet*. See PDF specification 8.2.4 “Collections” and 3.10.5 “Collection items” [3].

Usually Acrobat Reader 7 or 8 shows the embedded files in a table at the bottom with the following columns:

Name	Description	Modified	Size
...	...	...	...

If the files form a collection, then they are displayed in a table left or top (depending on option `view`, see `\embedfilesetup`).

Collection support is enabled automatically, if it is used.

`\embedfilesetup {<options>}`

The following options are supported in addition to options for `\embedfile`:

**view** If the PDF file contains a collection, then Acrobat Reader 8 shows a line at the top below the menu bar and the toolbar. It shows the current selected file, icons for changing the view mode, an options menu. The initial mode how the collection is presented is set by this option `view`. The following modes/values are supported, the default is `details`:

**details** The full collection table is displayed at the top below the collection bar.

**tile** The files of the collection are shown in tile mode on the left.

**hidden** The collection table is not shown.

**initialfile** Selects the file that is initially presented. Especially useful for an embedded PDF file that is then shown instead of the cover document.

`\embedfilefield {⟨key⟩} {⟨options⟩}`

Macro `\embedfilefield` defines a column/field in the collection table. The name of the field is `⟨key⟩`.

**type** sets the type of the field. The supported values are:

**text** A text field. Its value is set in `\embedfile` by option `⟨key⟩.value`.

**date** A date field. Its value is set in `\embedfile` by option `⟨key⟩.value`. A special format is required, see “3.8.3 Dates” [3].

**number** A field with an integer or float number. Its value is set in `\embedfile` by option `⟨key⟩.value`.

**file** The file name of the embedded file.

**desc** The description text of the embedded file. It is set in `\embedfile` by option `desc`.

**moddate** The modification date of the embedded file.

**size** The size of the embedded file.

All types allow the use of a prefix that is disregarded by sorting. The prefix for this field is set in `\embedfile` by option `⟨key⟩.prefix`.

**title** sets the column title.

**visible** controls whether the column is presented:

**true** shows the column.

**false** hides the column.

Default: **true**

**edit** Allows the editing of field values. Does not seem to have an effect for Acrobat Reader.

**true** enables the feature, if available (depends on the PDF viewer).

**false** disables the feature.

Default: **false**

The order of `\embedfilefield` statements defines the order of the columns.

`\embedfilesort {⟨key-sort-list⟩}`

The sort order of the embedded files are controlled by macro `\embedfilesort`. `⟨key-sort-list⟩` defines the sort order. The key is a field name defined by `\embedfilefield`. Its value is either **ascending** or **descending**. The default is **ascending**.

## 1.4 Examples

### 1.4.1 plain- $\TeX$

The package can be used with plain- $\TeX$ . It can be used with or without help from `miniltx.tex`.

If additionally package `keyval` (`graphicx`) is needed, load it first. Then package `embedfile` avoids a duplicate loading of package `keyval`.

Because plain-TeX does not provide a hook at end of the document, you have to call `\embedfilefinish` manually at the end after the last embedded file.

```

1 (*exampleplain)
2 % Load packages
3 \input miniltx
4 % \def\Gin@driver{pdftex.def}
5 % \input graphicx.sty
6 \input embedfile.sty
7 \resetatcatcode
8
9 % default setting
10 \embedfilesetup{
11   mimetype=text/plain
12 }
13
14 % Embed files
15 \embedfile[
16   filespec=example.tex,
17   desc={Source code (plain-TeX) of this example}
18 ]{embedfile-example-plain.tex}
19
20 \embedfile[
21   desc={Source of package 'embedfile'}
22 ]{embedfile.dtx}
23
24 \embedfile[
25   mimetype=application/pdf,
26   desc={Documentation of package 'embedfile'}
27 ]{embedfile.pdf}
28
29 % Some text
30 This example document contains three embedded files.
31
32 % End of document
33 \embedfilefinish % don't forget
34 \bye
35 /exampleplain)

```

#### 1.4.2 Collection example

```

36 (*examplecollection)
37 \NeedsTeXFormat{LaTeX2e}
38 \documentclass{article}
39 \usepackage[bookmarks=false]{hyperref}
40 % provides \pdfstringdef that is then used by 'title' and
41 % other keys.
42 \usepackage{embedfile}[2007/10/28]
43 \embedfilesetup{
44   view=details,
45   initialfile=embedfile.pdf
46 }
47 \embedfilefield{file}{
48   type=file,
49   title={File name}
50 }
51 \embedfilefield{description}{
52   type=desc,
53   title={Description}
54 }
55 \embedfilefield{date}{
56   type=moddate,
57   title={Date}

```

```

58 }
59 \embedfilefield{size}{
60   type=size,
61   title={Size}
62 }
63 \embedfilefield{type}{
64   type=text,
65   title={Type},
66   visible=false
67 }
68 \embedfilesort{
69   type,
70   date=descending
71 }
72 \begin{document}
73 An example for embedded files as collection.
74 You need Acrobat Reader 8 or higher.
75
76 \embedfile[
77   desc={Source file of package 'embedfile'},
78   description.prefix={Package: },
79   type.value={DTX}
80 ]{embedfile.dtx}
81
82 \embedfile[
83   desc={Documentation of package 'embedfile'},
84   description.prefix={Package: },
85   type.value={PDF}
86 ]{embedfile.pdf}
87
88 \embedfile[
89   desc={The source for this example},
90   description.prefix={Example: },
91   type.value={TEX}
92 ]{\jobname.tex}
93
94 \end{document}
95 %
96 \end{examplecollection}

```

## 1.5 Package dtx-attach

Package `dtx-attach` is just a small application of package `embedfile`. I am using it for the CTAN documentation of my packages in [CTAN:macros/latex/contrib/oberdiek/](https://ctan.org/ctan/ma/macros/latex/contrib/oberdiek/). It also serves as small example for the use of the package with L<sup>A</sup>T<sub>E</sub>X.

```

97 (*dtxattach)
98 \NeedsTeXFormat{LaTeX2e}
99 \ProvidesPackage{dtx-attach}
100 [2007/10/28 v2.0 Embed \string\jobname.dtx (HO)]%
101 \RequirePackage{embedfile}[2007/10/28]
102 \embedfile[%
103   stringmethod=escape,%
104   mimetype=plain/text,%
105   desc={LaTeX docstrip source archive for package '\jobname'}%
106 ]{\jobname.dtx}
107 \end{dtxattach}

```

## 2 Implementation

```

108 (*package)

```

## 2.1 Reload check and package identification

Reload check, especially if the package is not used with L<sup>A</sup>T<sub>E</sub>X.

```
109 \begingroup
110   \catcode44 12 % ,
111   \catcode45 12 % -
112   \catcode46 12 % .
113   \catcode58 12 % :
114   \catcode64 11 % @
115   \expandafter\let\expandafter\x\csname ver@embedfile.sty\endcsname
116   \ifcase 0%
117     \ifx\x\relax % plain
118     \else
119       \ifx\x\empty % LaTeX
120       \else
121         1%
122       \fi
123     \fi
124   \else
125     \expandafter\ifx\csname PackageInfo\endcsname\relax
126       \def\x#1#2{%
127         \immediate\write-1{Package #1 Info: #2.}%
128       }%
129     \else
130       \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
131     \fi
132     \x{embedfile}{The package is already loaded}%
133   \endgroup
134   \expandafter\endinput
135 \fi
136 \endgroup
```

Package identification:

```
137 \begingroup
138   \catcode40 12 % (
139   \catcode41 12 % )
140   \catcode44 12 % ,
141   \catcode45 12 % -
142   \catcode46 12 % .
143   \catcode47 12 % /
144   \catcode58 12 % :
145   \catcode64 11 % @
146   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
147     \def\x#1#2#3[#4]{\endgroup
148       \immediate\write-1{Package: #3 #4}%
149       \xdef#1{#4}%
150     }%
151   \else
152     \def\x#1#2[#3]{\endgroup
153       #2[#{#3}]%
154       \ifx#1\relax
155         \xdef#1{#3}%
156       \fi
157     }%
158   \fi
159   \expandafter\x\csname ver@embedfile.sty\endcsname
160   \ProvidesPackage{embedfile}%
161   [2007/10/28 v2.0 embed files into PDF (H0)]
```

## 2.2 Catcodes

```
162 \expandafter\edef\csname EmFi@AtEnd\endcsname{%
163   \catcode64 \the\catcode64\relax
```

```

164 }
165 \catcode64 11 % @
166 \def\TMP@EnsureCode#1#2{%
167   \edef\EmFi@AtEnd{%
168     \EmFi@AtEnd
169     \catcode#1 \the\catcode#1\relax
170   }%
171   \catcode#1 #2\relax
172 }
173 \TMP@EnsureCode{39}{12}% '
174 \TMP@EnsureCode{40}{12}% (
175 \TMP@EnsureCode{41}{12}% )
176 \TMP@EnsureCode{44}{12}% ,
177 \TMP@EnsureCode{46}{12}% .
178 \TMP@EnsureCode{47}{12}% /
179 \TMP@EnsureCode{58}{12}% :
180 \TMP@EnsureCode{60}{12}% <
181 \TMP@EnsureCode{61}{12}% =
182 \TMP@EnsureCode{62}{12}% >
183 \TMP@EnsureCode{91}{12}% [
184 \TMP@EnsureCode{93}{12}% ]
185 \TMP@EnsureCode{96}{12}% `

```

## 2.3 Tools

\EmFi@RequirePackage

```

186 \begingroup\expandafter\expandafter\expandafter\endgroup
187 \expandafter\ifx\csname RequirePackage\endcsname\relax
188   \def\EmFi@RequirePackage#1[#2]{%
189     \input #1.sty\relax
190   }%
191 \else
192   \let\EmFi@RequirePackage\RequirePackage
193 \fi

```

\EmFi@Error

```

194 \EmFi@RequirePackage{infwarerr}[2007/09/09]%
195 \def\EmFi@Error{%
196   \@PackageError{embedfile}%
197 }

```

## 2.4 Check for recent pdfTeX in PDF mode

Load package ifpdf and check mode.

```

198 \EmFi@RequirePackage{ifpdf}[2007/09/09]
199 \ifpdf
200 \else
201   \EmFi@Error{%
202     Missing pdfTeX in PDF mode%
203   }{%
204     Currently other drivers are not supported. %
205     Package loading is aborted.%
206   }%
207   \EmFi@AtEnd
208   \expandafter\endinput
209 \fi

```

Check version.

```

210 \begingroup\expandafter\expandafter\expandafter\endgroup
211 \expandafter\ifx\csname pdffilesize\endcsname\relax
212   \EmFi@Error{%
213     Unsupported pdfTeX version%
214   }{%

```



```

215     At least version 1.30 is necessary. Package loading is aborted.%
216 }%
217 \EmFi@AtEnd
218 \expandafter\endinput
219 \fi

```

## 2.5 Strings

Minimal version of package pdfescape is 2007/08/27 v1.5 because of \EdefSanitize.

```

220 \EmFi@RequirePackage{pdfescape}[2007/10/27]

221 \def\EmFi@temp#1{%
222   \expandafter\EdefSanitize\csname EmFi@S@#1\endcsname{#1}%
223 }

\EmFi@details
224 \EmFi@temp{details}%

\EmFi@tile
225 \EmFi@temp{tile}%

\EmFi@hidden
226 \EmFi@temp{hidden}%

\EmFi@S@text
227 \EmFi@temp{text}

\EmFi@S@date
228 \EmFi@temp{date}

\EmFi@S@number
229 \EmFi@temp{number}

\EmFi@S@file
230 \EmFi@temp{file}

\EmFi@S@desc
231 \EmFi@temp{desc}

\EmFi@S@moddate
232 \EmFi@temp{moddate}

\EmFi@S@creationdate
233 \EmFi@temp{creationdate}

\EmFi@S@size
234 \EmFi@temp{size}

\EmFi@S@ascending
235 \EmFi@temp{ascending}

\EmFi@S@descending
236 \EmFi@temp{descending}

\EmFi@S@true
237 \EmFi@temp{true}

\EmFi@S@false
238 \EmFi@temp{false}

```

## 2.6 Switches

```

\ifEmFi@collection
239 \newif\ifEmFi@collection

\ifEmFi@initialfile
240 \newif\ifEmFi@initialfile

\ifEmFi@sort
241 \newif\ifEmFi@sort

\ifEmFi@visible
242 \newif\ifEmFi@visible

\ifEmFi@edit
243 \newif\ifEmFi@edit

\ifEmFi@item
244 \newif\ifEmFi@item

\ifEmFi@finished
245 \newif\ifEmFi@finished

```

## 2.7 Key value definitions

```

246 \expandafter\ifx\csname define@key\endcsname\relax
247   \chardef\EmFi@plain=\z@
248   \def\EmFi@temp#1{%
249     \begingroup\expandafter\expandafter\expandafter\endgroup
250     \expandafter\ifx\csname#1\endcsname\relax
251       \chardef\EmFi@plain=\@ne
252       \fi
253   }%
254   \EmFi@temp{NeedsTeXFormat}%
255   \EmFi@temp{ProvidesPackage}%
256   \EmFi@temp{DeclareOption}%
257   \EmFi@temp{ExecuteOptions}%
258   \EmFi@temp{ProcessOptions}%
259   \ifnum\EmFi@plain=\@ne
260     \def\EmFi@temp#1{%
261       \expandafter\let\csname EmFi@Org#1\expandafter\endcsname
262         \csname#1\endcsname
263       \expandafter\def\csname#1\endcsname
264     }%
265     \EmFi@temp{NeedsTeXFormat}#1{}%
266     \EmFi@temp{ProvidesPackage}#1[#2]{}% hash-ok
267     \EmFi@temp{DeclareOption}#1{}%
268     \EmFi@temp{ExecuteOptions}#1{}%
269     \EmFi@temp{ProcessOptions}{}%

\KV@errx LATEX's option processing is not available with plain-TEX. Thus we define the de-
fault error command \KV@errx here, also using package infwarerr's \@PackageError.

270   \def\KV@errx#1{%
271     \@PackageError{keyval}{#1}\@ehc
272   }%

```

Other macros from L<sup>A</sup>T<sub>E</sub>X's kernel that are used by package keyval.

\@ifnextchar

```
273 \expandafter\ifx\csname @ifnextchar\endcsname\relax
274 \def\@ifnextchar#1#2#3{%
275   \let\reserved@d=#1%
276   \def\reserved@a{#2}%
277   \def\reserved@b{#3}%
278   \futurelet\@let@token\@ifnch
279 }%
280 \def\@ifnch{%
281   \ifx\@let@token\@sptoken
282     \let\reserved@c\@xifnch
283   \else
284     \ifx\@let@token\reserved@d
285       \let\reserved@c\reserved@a
286     \else
287       \let\reserved@c\reserved@b
288     \fi
289   \fi
290   \reserved@c
291 }%
292 \begingroup
293 \def\:{\global\let\@sptoken= }%
294 \: % this makes \@sptoken a space token
295 \def\:{\@xifnch}%
296 \expandafter\gdef\:{%
297   \futurelet\@let@token\@ifnch
298 }%
299 \endgroup
300 \fi
```

\@namedef

```
301 \expandafter\ifx\csname @namedef\endcsname\relax
302 \def\@namedef#1{%
303   \expandafter\def\csname#1\endcsname
304 }%
305 \fi
306 \fi
307 \EmFi@RequirePackage{keyval}[1999/03/16]%
308 \ifnum\EmFi@plain=\@ne
309 \def\EmFi@temp#1{%
310   \expandafter\let\csname#1\expandafter\endcsname
311   \csname EmFi@Org#1\endcsname
312 }%
313 \EmFi@temp{NeedsTeXFormat}%
314 \EmFi@temp{ProvidesPackage}%
315 \EmFi@temp{DeclareOption}%
316 \EmFi@temp{ExecuteOptions}%
317 \EmFi@temp{ProcessOptions}%
318 \fi
319 \fi
```

\EmFi@GlobalKey

```
320 \def\EmFi@GlobalKey#1#2{%
321   \global\expandafter\let\csname KV@#1@#2\expandafter\endcsname
322   \csname KV@#1@#2\endcsname
323 }
```

\EmFi@GlobalDefaultKey

```
324 \def\EmFi@GlobalDefaultKey#1#2{%
325   \EmFi@GlobalKey{#1}{#2}%
326   \global\expandafter\let
```

```

327     \csname KV@#1@#2@default\expandafter\endcsname
328     \csname KV@#1@#2@default\endcsname
329 }

```

\EmFi@DefineKey

```

330 \def\EmFi@DefineKey#1#2{%
331   \define@key{EmFi}{#1}{%
332     \expandafter\def\csname EmFi@#1\endcsname{##1}%
333   }%
334   \expandafter\def\csname EmFi@#1\endcsname{#2}%
335 }

```

Subtype of the embedded file (optional).

```
336 \EmFi@DefineKey{mimetype}{}
```

File specification string.

```
337 \EmFi@DefineKey{filespec}{\EmFi@file}
```

File system (optional).

```
338 \EmFi@DefineKey{filesystem}{}
```

Description (optional).

```
339 \EmFi@DefineKey{desc}{}
```

Method for converting text to PDF strings.

```

340 \EmFi@DefineKey{stringmethod}{%
341   \ifx\pdfstringdef\@undefined
342     escape%
343   \else
344     \ifx\pdfstringdef\relax
345       escape%
346     \else
347       psd%
348     \fi
349   \fi
350 }

```

Initial view of the collection.

```

351 \define@key{EmFi}{view}[]{%
352   \EdefSanitize\EmFi@temp{#1}%
353   \def\EmFi@next{%
354     \global\EmFi@collectiontrue
355   }%
356   \ifx\EmFi@temp\empty
357     \let\EmFi@view\EmFi@S@details
358   \else\ifx\EmFi@temp\EmFi@S@details
359     \let\EmFi@view\EmFi@S@details
360   \else\ifx\EmFi@temp\EmFi@S@tile
361     \let\EmFi@view\EmFi@S@tile
362   \else\ifx\EmFi@temp\EmFi@S@hidden
363     \let\EmFi@view\EmFi@S@hidden
364   \else
365     \let\EmFi@next\relax
366     \EmFi@Error{%
367       Unknown value '\EmFi@temp' for key 'view'.\MessageBreak
368       Supported values: 'details', 'tile', 'hidden'.%
369     }\@ehc
370   \fi\fi\fi\fi
371   \EmFi@next
372 }
373 \EmFi@DefineKey{initialfile}{}
```

```

\embedfilesetup
374 \def\embedfilesetup{%
375   \ifEmFi@finished
376     \def\EmFi@next##1{%
377       \EmFi@Error{%
378         \string\embedfilefield\space after \string\embedfilefinish
379       }{%
380         The list of embedded files is already written.%
381       }%
382     \else
383       \def\EmFi@next{%
384         \setkeys{EmFi}%
385       }%
386     \fi
387     \EmFi@next
388 }

\EmFi@schema
389 \def\EmFi@schema{}

\EmFi@order
390 \gdef\EmFi@order{0}

\EmFi@@order
391 \let\EmFi@@order\relax

\EmFi@fieldlist
392 \def\EmFi@fieldlist{}

\EmFi@sortcase
393 \def\EmFi@sortcase{0}%

\embedfilefield
394 \def\embedfilefield#1#2{%
395   \ifEmFi@finished
396     \EmFi@Error{%
397       \string\embedfilefield\space after \string\embedfilefinish
398     }{%
399       The list of embedded files is already written.%
400     }%
401   \else
402     \global\EmFi@collectiontrue
403     \edefSanitize\EmFi@key{#1}%
404     \expandafter\ifx\csname KV@EmFi@\EmFi@key.prefix\endcsname\relax
405       \begingroup
406         \count@=\EmFi@order
407         \advance\count@ 1 %
408         \xdef\EmFi@order{\the\count@}%
409         \let\EmFi@title\EmFi@key
410         \let\EmFi@type\EmFi@S@text
411         \EmFi@visibletrue
412         \EmFi@editfalse
413         \setkeys{EmFiFi}{#2}%
414         \EmFi@convert\EmFi@title\EmFi@title
415         \xdef\EmFi@schema{%
416           \EmFi@schema
417           /\pdfescapename{\EmFi@key}<<%
418           /Subtype/%
419           \ifx\EmFi@type\EmFi@S@date D%
420           \else\ifx\EmFi@type\EmFi@S@number N%
421           \else\ifx\EmFi@type\EmFi@S@file F%

```

```

422         \else\ifx\EmFi@type\EmFi@S@desc Desc%
423         \else\ifx\EmFi@type\EmFi@S@moddate ModDate%
424         \else\ifx\EmFi@type\EmFi@S@creationdate CreationDate%
425         \else\ifx\EmFi@type\EmFi@S@size Size%
426         \else S%
427         \fi\fi\fi\fi\fi\fi\fi
428         /N(\EmFi@title)%
429         \EmFi@@order{\EmFi@order}%
430         \ifEmFi@visible
431         \else
432             /V false%
433         \fi
434         \ifEmFi@edit
435             /E true%
436         \fi
437     >>%
438 }%
439 \let\do\relax
440 \xdef\EmFi@fieldlist{%
441     \EmFi@fieldlist
442     \do{\EmFi@key}%
443 }%
444 \ifx\EmFi@type\EmFi@S@text
445     \define@key{EmFi}{\EmFi@key.value}{%
446         \EmFi@itemtrue
447         \def\EmFi@temp{##1}%
448         \EmFi@convert\EmFi@temp\EmFi@temp
449         \expandafter\def\csname EmFi@V@#1%
450         \expandafter\endcsname\expandafter{%
451             \expandafter(\EmFi@temp)%
452         }%
453     }%
454     \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
455 \else\ifx\EmFi@type\EmFi@S@date
456     \define@key{EmFi}{\EmFi@key.value}{%
457         \EmFi@itemtrue
458         \def\EmFi@temp{##1}%
459         \EmFi@convert\EmFi@temp\EmFi@temp
460         \expandafter\def\csname EmFi@V@#1%
461         \expandafter\endcsname\expandafter{%
462             \expandafter(\EmFi@temp)%
463         }%
464     }%
465     \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
466 \else\ifx\EmFi@type\EmFi@S@number
467     \define@key{EmFi}{\EmFi@key.value}{%
468         \EmFi@itemtrue
469         \expandafter\edef\csname EmFi@V@#1\endcsname{ ##1}%
470     }%
471     \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
472 \fi\fi\fi
473 \define@key{EmFi}{\EmFi@key.prefix}{%
474     \EmFi@itemtrue
475     \expandafter\def\csname EmFi@P@#1\endcsname{##1}%
476 }%
477 \EmFi@GlobalKey{EmFi}{\EmFi@key.prefix}%
478 \define@key{EmFiSo}{\EmFi@key}[ascending]{%
479     \edef\Sanitize\EmFi@temp{##1}%
480     \ifx\EmFi@temp\EmFi@S@ascending
481         \def\EmFi@temp{true}%
482     \else\ifx\EmFi@temp\EmFi@S@descending
483         \def\EmFi@temp{false}%

```

```

484         \else
485         \def\EmFi@temp{}%
486         \EmFi@Error{%
487             Unknown sort order '\EmFi@temp'.\MessageBreak
488             Supported values: '\EmFi@S@ascending', %
489             '\EmFi@S@descending
490         }\@ehc
491     \fi\fi
492     \ifx\EmFi@temp\empty
493     \else
494         \xdef\EmFi@sortkeys{%
495             \EmFi@sortkeys
496             /\pdfescapename{#1}%
497         }%
498         \ifx\EmFi@sortorders\empty
499             \global\let\EmFi@sortorders\EmFi@temp
500             \gdef\EmFi@sortcase{1}%
501         \else
502             \xdef\EmFi@sortorders{%
503                 \EmFi@sortorders
504                 \space
505                 \EmFi@temp
506             }%
507             \xdef\EmFi@sortcase{2}%
508         \fi
509     \fi
510     }%
511     \EmFi@GlobalDefaultKey{EmFiSo}\EmFi@key
512 \endgroup
513 \else
514     \EmFi@Error{%
515         Field '\EmFi@key' is already defined%
516     }\@ehc
517 \fi
518 \fi
519 }

520 \define@key{EmFiFi}{type}{%
521     \EdefSanitize\EmFi@temp{#1}%
522     \ifx\EmFi@temp\EmFi@S@text
523         \let\EmFi@type\EmFi@temp
524     \else\ifx\EmFi@temp\EmFi@S@date
525         \let\EmFi@type\EmFi@temp
526     \else\ifx\EmFi@temp\EmFi@S@number
527         \let\EmFi@type\EmFi@temp
528     \else\ifx\EmFi@temp\EmFi@S@file
529         \let\EmFi@type\EmFi@temp
530     \else\ifx\EmFi@temp\EmFi@S@desc
531         \let\EmFi@type\EmFi@temp
532     \else\ifx\EmFi@temp\EmFi@S@moddate
533         \let\EmFi@type\EmFi@temp
534     \else\ifx\EmFi@temp\EmFi@S@creationdate
535         \let\EmFi@type\EmFi@temp
536     \else\ifx\EmFi@temp\EmFi@S@size
537         \let\EmFi@type\EmFi@temp
538     \else
539         \EmFi@Error{%
540             Unknown type '\EmFi@temp'.\MessageBreak
541             Supported types: 'text', 'date', 'number', 'file',\MessageBreak
542             'desc', 'moddate', 'creationdate', 'size'%
543         }%
544     \fi\fi\fi\fi\fi\fi\fi
545 }

```

```

546 \define@key{EmFiFi}{title}{%
547   \def\EmFi@title{#1}%
548 }

\EmFi@setboolean

549 \def\EmFi@setboolean#1#2{%
550   \edefSanitize\EmFi@temp{#2}%
551   \ifx\EmFi@temp\EmFi@S@true
552     \csname EmFi@#1true\endcsname
553   \else
554     \ifx\EmFi@temp\EmFi@S@false
555       \csname EmFi@#1false\endcsname
556     \else
557       \EmFi@Error{%
558         Unknown value '\EmFi@temp' for key '#1'.\MessageBreak
559         Supported values: 'true', 'false'%
560       }\@ehc
561     \fi
562   \fi
563 }

564 \define@key{EmFiFi}{visible}[true]{%
565   \EmFi@setboolean{visible}{#1}%
566 }

567 \define@key{EmFiFi}{edit}[true]{%
568   \EmFi@setboolean{edit}{#1}%
569 }

\EmFi@sortkeys

570 \def\EmFi@sortkeys{}

\EmFi@sortorders

571 \def\EmFi@sortorders{}

\embedfilesort

572 \def\embedfilesort{%
573   \setkeys{EmFiSo}%
574 }

```

## 2.8 Embed the file

```

\embedfile

575 \def\embedfile{%
576   \@ifnextchar[\EmFi@embedfile{\EmFi@embedfile[]}%
577 }

\EmFi@embedfile

578 \def\EmFi@embedfile[#1]#2{%
579   \ifEmFi@finished
580     \EmFi@Error{%
581       \string\embedfile\space after \string\embedfilefinish
582     }{%
583       The list of embedded files is already written.%
584     }%
585   \else
586     \begingroup
587     \def\EmFi@file{#2}%
588     \ifx\EmFi@file\EmFi@initialfile
589       \global\EmFi@initialfiletrue
590     \fi

```



```

591 \setkeys{EmFi}{#1}%
592 \expandafter\ifx\expandafter\\pdffilesize{\EmFi@file}\\%
593 \EmFi@Error{%
594   File '\EmFi@file' not found%
595 }{%
596   The unknown file is not embedded.%
597 }%
598 \else
599 \EmFi@convert\EmFi@filespec\EmFi@@filespec
600 \ifx\EmFi@desc\empty
601   \let\EmFi@@desc\empty
602 \else
603   \EmFi@convert\EmFi@desc\EmFi@@desc
604 \fi
605 \ifEmFi@item
606   \let\do\EmFi@do
607   \immediate\pdfobj{%
608     <<%
609       \EmFi@fieldlist
610     >>%
611   }%
612   \edef\EmFi@ci{\the\pdflastobj}%
613 \fi
614 \immediate\pdfobj stream attr{%
615   /Type/EmbeddedFile%
616   \ifx\EmFi@mimetype\empty
617   \else
618     /Subtype/\pdfescapename{\EmFi@mimetype}%
619   \fi
620   /Params<<%
621     /ModDate(\pdffilemoddate{\EmFi@file})%
622     /Size \pdffilesize{\EmFi@file}%
623     /Checksum<\pdfmdfivesum file{\EmFi@file}>%
624   >>%
625 }file{\EmFi@file}\relax
626 \immediate\pdfobj{%
627   <<%
628     /Type/Filespec%
629     \ifx\EmFi@filesystem\empty
630     \else
631       /FS/\pdfescapename{\EmFi@filesystem}%
632     \fi
633     /F(\EmFi@@filespec)%
634     \ifx\EmFi@@desc\empty
635     \else
636       /Desc(\EmFi@@desc)%
637     \fi
638     /EF<<%
639       /F \the\pdflastobj\space 0 R%
640     >>%
641     \ifEmFi@item
642       /CI \EmFi@ci\space 0 R%
643     \fi
644   >>%
645 }%
646 \EmFi@add{%
647   \EmFi@@filespec
648 }{\the\pdflastobj\space 0 R}%
649 \fi
650 \endgroup
651 \fi
652 }

```

\EmFi@do

```
653 \def\EmFi@do#1{%
654   \expandafter\ifx\csname EmFi@P@#1\endcsname\relax
655     \expandafter\ifx\csname EmFi@V@#1\endcsname\relax
656       \else
657         /\pdfescapename{#1}\csname EmFi@V@#1\endcsname
658       \fi
659     \else
660       /\pdfescapename{#1}<<%
661       \expandafter\ifx\csname EmFi@V@#1\endcsname\relax
662         \else
663           /D\csname EmFi@V@#1\endcsname
664         \fi
665       /P(\csname EmFi@P@#1\endcsname)%
666     >>%
667   \fi
668 }
```

\EmFi@convert

```
669 \def\EmFi@convert#1#2{%
670   \ifnum\pdfstrcmp{\EmFi@stringmethod}{psd}=0 %
671     \pdfstringdef\EmFi@temp{#1}%
672     \let#2\EmFi@temp
673   \else
674     \edef#2{\pdfescapestring{#1}}%
675   \fi
676 }
```

```
677 \global\let\EmFi@list\empty
```

\EmFi@add Sorting is done by the insertion sort algorithm. Probably the sorting could be done more reliable. However, the PDF specification is not too clear to me regarding precise sorting rules (how to deal with different encodings, escaped characters, ...).

```
678 \def\EmFi@add#1#2{%
679   \begingroup
680   \edef\key{\pdfescapehex{#1}}%
681   \ifx\EmFi@list\empty
682     \xdef\EmFi@list{\noexpand\do{\key}{#2}}%
683   \else
684     \def\do##1##2{%
685       \ifnum\pdfstrcmp{##1}{\key}>0 %
686         \edef\x{%
687           \toks@{%
688             \the\toks@%
689             \noexpand\do{\key}{#2}%
690             \noexpand\do{##1}{##2}%
691           }%
692         }%
693         \x
694       \def\do####1####2{%
695         \toks@\expandafter{\the\toks@\do{####1}{####2}}%
696       }%
697       \def\stop{%
698         \xdef\EmFi@list{\the\toks@}%
699       }%
700     \else
701       \toks@\expandafter{\the\toks@\do{##1}{##2}}%
702     \fi
703   }%
704   \def\stop{%
705     \xdef\EmFi@list{\the\toks@\noexpand\do{\key}{#2}}%
```

```

706     }%
707     \toks@{ }%
708     \EmFi@list\stop
709     \fi
710 \endgroup
711 }

\embedfilefinish

712 \def\embedfilefinish{%
713   \ifEmFi@finished
714     \EmFi@Error{%
715       Too many invocations of \string\embedfilefinish
716     }{%
717       The list of embedded files is already written.%
718     }%
719   \else
720     \ifx\EmFi@list\empty
721     \else
Write /EmbeddedFiles entry.
722       \global\EmFi@finishedtrue
723       \begingroup
724       \def\do##1##2{%
725         <##1>##2%
726       }%
727       \immediate\pdfobj{%
728         <<%
729           /Names[\EmFi@list]%
730         >>%
731       }%
732       \pdfnames{%
733         /EmbeddedFiles \the\pdflastobj\space 0 R%
734       }%
735     \endgroup
Write collection objects.
736     \ifEmFi@initialfile
737     \EmFi@collectiontrue
738     \fi
739     \ifEmFi@collection
740     \ifEmFi@initialfile
741     \else
742       \ifx\EmFi@initialfile\empty
743       \EmFi@convert\EmFi@initialfile\EmFi@initialfile
744     \else
745       \@PackageWarningNoLine{embedfile}{%
746         Missing initial file '\EmFi@initialfile'\MessageBreak
747         among the embedded files%
748       }%
749       \EmFi@initialfilefalse
750     \fi
751   \fi
752   \ifcase\EmFi@sortcase
753     \def\EmFi@temp{ }%
754   \or
755     \def\EmFi@temp{%
756       /S\EmFi@sortkeys
757       /A \EmFi@sortorders
758     }%
759   \else
760     \def\EmFi@temp{%
761       /S[\EmFi@sortkeys]%
762       /A[\EmFi@sortorders]%

```

```

763     }%
764     \fi
765     \def\EmFi@@order##1{%
766         \ifnum\EmFi@order>1 %
767             /O ##1%
768         \fi
769     }%
770     \immediate\pdfobj{%
771         <<%
772         \ifx\EmFi@schema\empty
773             \else
774                 /Schema<<\EmFi@schema>>%
775             \fi
776             \ifEmFi@initialfile
777                 /D(\EmFi@initialfile)%
778             \fi
779             \ifx\EmFi@view\EmFi@S@tile
780                 /View/T%
781             \else\ifx\EmFi@view\EmFi@S@hidden
782                 /View/H%
783             \fi\fi
784             \ifx\EmFi@temp\empty
785                 \EmFi@temp
786             \else
787                 /Sort<<\EmFi@temp>>%
788             \fi
789         >>%
790     }%
791     \pdfcatalog{%
792         /Collection \the\pdflastobj\space0 R%
793     }%
794     \fi
795     \fi
796     \fi
797 }

798 \begingroup\expandafter\expandafter\expandafter\endgroup
799 \expandafter\ifx\csname AtEndDocument\endcsname\relax
800 \else
801     \AtEndDocument{\embedfilefinish}%
802 \fi

803 \EmFi@AtEnd
804 \end{package}

```

## 3 Test

### 3.1 Catcode checks for loading

```

805 \test1\
806 \catcode'\@=11 %
807 \def\RestoreCatcodes{}
808 \count@=0 %
809 \loop
810     \edef\RestoreCatcodes{%
811         \RestoreCatcodes
812         \catcode\the\count@=\the\catcode\count@\relax
813     }%
814     \ifnum\count@<255 %
815         \advance\count@\@ne
816     \repeat
817

```

```

818 \def\RangeCatcodeInvalid#1#2{%
819   \count@=#1\relax
820   \loop
821     \catcode\count@=15 %
822     \ifnum\count@<#2\relax
823       \advance\count@\@ne
824     \repeat
825 }
826 \def\Test{%
827   \RangeCatcodeInvalid{0}{47}%
828   \RangeCatcodeInvalid{58}{64}%
829   \RangeCatcodeInvalid{91}{96}%
830   \RangeCatcodeInvalid{123}{255}%
831   \catcode'\@=12 %
832   \catcode'\=0 %
833   \catcode'\{=1 %
834   \catcode'\}=2 %
835   \catcode'\#=6 %
836   \catcode'\[=12 %
837   \catcode'\]=12 %
838   \catcode'\%=14 %
839   \catcode'\_ =10 %
840   \catcode13=5 %
841   \input embedfile.sty\relax
842   \RestoreCatcodes
843 }
844 \Test
845 \csname @@end\endcsname
846 \end
847 </test1>

```

## 3.2 Simple test

```

848 <*test2>
849 \input embedfile.sty\relax
850 \embedfile[%
851   stringmethod=escape,%
852   mimetype=plain/text,%
853   desc={LaTeX docstrip source archive for package 'embedfile'}%
854 ]{embedfile.dtx}
855 \nopagenumbers
856 Test (plain-TeX): {\tt embedfile.dtx} should be embedded.%
857 \embedfilefinish
858 \bye
859 </test2>

860 <*test3>
861 \NeedsTeXFormat{LaTeX2e}
862 \let\SavedJobname\jobname
863 \def\jobname{embedfile}
864 \RequirePackage{dtx-attach}[2007/10/28]
865 \let\jobname\SavedJobname
866 \documentclass{minimal}
867 \begin{document}
868   Test (\LaTeX): \texttt{embedfile.dtx} should be embedded.%
869 \end{document}
870 </test3>

```

## 4 Installation

### 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

---

<sup>1</sup><ftp://ftp.ctan.org/tex-archive/>

[CTAN:macros/latex/contrib/oberdiek/embedfile.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/embedfile.pdf](#) Documentation.

**Bundle.** All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

[CTAN:macros/latex/contrib/oberdiek/oberdiek-tds.zip](#)

TDS refers to the standard “A Directory Structure for T<sub>E</sub>X Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

## 4.2 Bundle installation

**Unpacking.** Unpack the `oberdiek-tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek-tds.zip -d ~/texmf
```

**Script installation.** Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

## 4.3 Package installation

**Unpacking.** The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain-T<sub>E</sub>X:

```
tex embedfile.dtx
```

**TDS.** Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

<code>embedfile.sty</code>	→ <code>tex/latex/oberdiek/embedfile.sty</code>
<code>dtx-attach.sty</code>	→ <code>tex/latex/oberdiek/dtx-attach.sty</code>
<code>embedfile.pdf</code>	→ <code>doc/latex/oberdiek/embedfile.pdf</code>
<code>embedfile-example-plain.tex</code>	→ <code>doc/latex/oberdiek/embedfile-example-plain.tex</code>
<code>embedfile-example-collection.tex</code>	→ <code>doc/latex/oberdiek/embedfile-example-collection.tex</code>
<code>test/embedfile-test1.tex</code>	→ <code>doc/latex/oberdiek/test/embedfile-test1.tex</code>
<code>test/embedfile-test2.tex</code>	→ <code>doc/latex/oberdiek/test/embedfile-test2.tex</code>
<code>test/embedfile-test3.tex</code>	→ <code>doc/latex/oberdiek/test/embedfile-test3.tex</code>
<code>embedfile.dtx</code>	→ <code>source/latex/oberdiek/embedfile.dtx</code>

If you have a `docstrip.cfg` that configures and enables `docstrip`’s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

## 4.4 Refresh file name databases

If your T<sub>E</sub>X distribution (teT<sub>E</sub>X, miK<sub>T</sub>E<sub>X</sub>, ...) relies on file name databases, you must refresh these. For example, teT<sub>E</sub>X users run `texhash` or `mktextlsr`.

## 4.5 Some details for the interested

**Attached source.** The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk embedfile.pdf unpack_files output .
```

**Unpacking with L<sup>A</sup>T<sub>E</sub>X.** The `.dtx` chooses its action depending on the format:  
**plain-T<sub>E</sub>X:** Run `docstrip` and extract the files.

**L<sup>A</sup>T<sub>E</sub>X:** Generate the documentation.

If you insist on using L<sup>A</sup>T<sub>E</sub>X for `docstrip` (really, `docstrip` does not need L<sup>A</sup>T<sub>E</sub>X), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{embedfile.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

**Generating the documentation.** You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL<sup>A</sup>T<sub>E</sub>X:

```
pdflatex embedfile.dtx
makeindex -s gind.ist embedfile.idx
pdflatex embedfile.dtx
makeindex -s gind.ist embedfile.idx
pdflatex embedfile.dtx
```

## 5 References

- [1] Scott Pakin: *The attachfile package*; 2005/02/20 v1.2; [CTAN:macros/latex/contrib/attachfile/](http://CTAN:macros/latex/contrib/attachfile/).
- [2] Heiko Oberdiek: *The attachfile2 package*; 2006/08/16 v2.2; [CTAN:macros/latex/contrib/oberdiek/attachfile2.pdf](http://CTAN:macros/latex/contrib/oberdiek/attachfile2.pdf).
- [3] Adobe Systems Incorporated: *PDF Reference, Sixth Edition, Version 1.7*, Oktober 2006; [http://www.adobe.com/devnet/pdf/pdf\\_reference.html](http://www.adobe.com/devnet/pdf/pdf_reference.html).
- [4] Network Working Group: RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, November 1996; <http://www.rfc-editor.org/>.
- [5] IANA (Internet Assigned Numbers Authority): *MIME Media Types*, May 2006; <http://www.iana.org/assignments/media-types/>.

## 6 History

[2006/08/16 v1.0]

- First public version.

[2007/04/11 v1.1]

- Line ends sanitized.

[2007/09/09 v1.2]

- Fixes for plain-T<sub>E</sub>X, wrapper for package `keyval` added.
- Catcode section rewritten.

[2007/10/28 v2.0]

- Collection support added (PDF 1.7).

## 7 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	E
\# ..... 835	\EdefSanitize ..... ..... 222, 352, 403, 469, 479, 521, 550
\% ..... 838	\embedfile ..... 2, 15, ..... 20, 24, 76, 82, 88, 102, <u>575</u> , 581, 850
\: ..... 293, 294, 295, 296	\embedfilefield ..... ..... 4, 47, 51, 55, 59, 63, 378, <u>394</u>
\@ ..... 806, 831	\embedfilefinish ..... 3, ..... 33, 378, 397, 581, <u>712</u> , 801, 857
\@PackageError ..... 196, 271	\embedfilessetup ..... 3, 3, 10, 43, <u>374</u>
\@PackageWarningNoLine ..... 745	\embedfilesort ..... 4, 68, <u>572</u>
\@ehc ..... 271, 369, 490, 516, 560	\EmFi@desc ..... 601, 603, 634, 636
\@ifnch ..... 278, 280, 297	\EmFi@filespec ..... 599, 633, 647
\@ifnextchar ..... <u>273</u> , 576	\EmFi@order ..... <u>391</u> , 429, 765
\@let@token ..... 278, 281, 284, 297	\EmFi@add ..... 646, <u>678</u>
\@namedef ..... <u>301</u>	\EmFi@AtEnd ... 167, 168, 207, 217, 803
\@ne ..... 251, 259, 308, 815, 823	\EmFi@ci ..... 612, 642
\@sptoken ..... 281, 293, 294	\EmFi@collectiontrue .. 354, 402, 737
\@undefined ..... 341	\EmFi@convert ..... ..... 414, 448, 459, 599, 603, <u>669</u> , 743
\@xifnch ..... 282, 295	\EmFi@DefineKey ..... ..... <u>330</u> , 336, 337, 338, 339, 340, 373
\[ ..... 836	\EmFi@desc ..... 600, 603
\\ ..... 592, 832	\EmFi@details ..... <u>224</u>
\{ ..... 833	\EmFi@do ..... 606, <u>653</u>
\} ..... 834	\EmFi@editfalse ..... 412
\] ..... 837	\EmFi@embedfile ..... 576, <u>578</u>
	\EmFi@Error ..... ..... 194, 201, 212, 366, 377, 396, ..... 486, 514, 539, 557, 580, 593, 714
\_ ..... 839	\EmFi@fieldlist ... <u>392</u> , 440, 441, 609
	\EmFi@file ..... 337, 587, ..... 588, 592, 594, 621, 622, 623, 625
<b>A</b>	\EmFi@filespec ..... 599
\advance ..... 407, 815, 823	\EmFi@filesystem ..... 629, 631
\AtEndDocument ..... 801	\EmFi@finishedtrue ..... 722
	\EmFi@GlobalDefaultKey .... <u>324</u> , 511
<b>B</b>	\EmFi@GlobalKey ..... ..... <u>320</u> , 325, 454, 465, 471, 477
\begin ..... 72, 867	\EmFi@hidden ..... <u>226</u>
\bye ..... 34, 858	\EmFi@initialfile ..... ..... 588, 742, 743, 746, 777
	\EmFi@initialfilefalse ..... 749
<b>C</b>	\EmFi@initialfiletrue ..... 589
\catcode .. 110, 111, 112, 113, 114, ..... 138, 139, 140, 141, 142, 143, ..... 144, 145, 163, 165, 169, 171, ..... 806, 812, 821, 831, 832, 833, ..... 834, 835, 836, 837, 838, 839, 840	\EmFi@itemtrue .... 446, 457, 468, 474
\chardef ..... 247, 251	\EmFi@key ..... 403, 404, 409, ..... 417, 442, 445, 454, 456, 465, ..... 467, 471, 473, 477, 478, 511, 515
\count@ ..... 406, 407, 408, 808, ..... 812, 814, 815, 819, 821, 822, 823	\EmFi@list ..... 677, ..... 681, 682, 698, 705, 708, 720, 729
\csname ..... 115, 125, ..... 146, 159, 162, 187, 211, 222, ..... 246, 250, 261, 262, 263, 273, ..... 301, 303, 310, 311, 321, 322, ..... 327, 328, 332, 334, 404, 449, ..... 460, 469, 475, 552, 555, 654, ..... 655, 657, 661, 663, 665, 799, 845	\EmFi@mimetype ..... 616, 618
	\EmFi@next 353, 365, 371, 376, 383, 387
<b>D</b>	\EmFi@order ... <u>390</u> , 406, 408, 429, 766
\define@key .... 331, 351, 445, 456, ..... 467, 473, 478, 520, 546, 564, 567	\EmFi@plain ..... 247, 251, 259, 308
\do ..... 439, 442, 606, 682, 684, ..... 689, 690, 694, 695, 701, 705, 724	\EmFi@RequirePackage ..... ..... <u>186</u> , 194, 198, 220, 307
\documentclass ..... 38, 866	



\EmFi@S@ascending	235, 480, 488	\Gin@driver	4
\EmFi@S@creationdate	233, 424, 534		
\EmFi@S@date	228, 419, 455, 524	I	
\EmFi@S@desc	231, 422, 530	\ifcase	116, 752
\EmFi@S@descending	236, 482, 489	\ifEmFi@collection	239, 739
\EmFi@S@details	357, 358, 359	\ifEmFi@edit	243, 434
\EmFi@S@false	238, 554	\ifEmFi@finished	245, 375, 395, 579, 713
\EmFi@S@file	230, 421, 528	\ifEmFi@initialfile	240, 736, 740, 776
\EmFi@S@hidden	362, 363, 781	\ifEmFi@item	244, 605, 641
\EmFi@S@moddate	232, 423, 532	\ifEmFi@sort	241
\EmFi@S@number	229, 420, 466, 526	\ifEmFi@visible	242, 430
\EmFi@S@size	234, 425, 536	\ifnum	259, 308, 670, 685, 766, 814, 822
\EmFi@S@text	227, 410, 444, 522	\ifpdf	199
\EmFi@S@tile	360, 361, 779	\ifx	117, 119, 125, 146, 154, 187, 211, 246, 250, 273, 281, 284, 301, 341, 344, 356, 358, 360, 362, 404, 419, 420, 421, 422, 423, 424, 425, 444, 455, 466, 480, 482, 492, 498, 522, 524, 526, 528, 530, 532, 534, 536, 551, 554, 588, 592, 600, 616, 629, 634, 654, 655, 661, 681, 720, 742, 772, 779, 781, 784, 799
\EmFi@S@true	237, 551	\immediate	127, 148, 607, 614, 626, 727, 770
\EmFi@schema	389, 415, 416, 772, 774	\input	3, 5, 6, 189, 841, 849
\EmFi@setboolean	549, 565, 568	J	
\EmFi@sortcase	393, 500, 507, 752	\jobname	92, 100, 105, 106, 862, 863, 865
\EmFi@sortkeys	494, 495, 570, 756, 761	K	
\EmFi@sortorders	498, 499, 502, 503, 571, 757, 762	\key	680, 682, 685, 689, 705
\EmFi@stringmethod	670	\KV@errx	270
\EmFi@temp	221, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 248, 254, 255, 256, 257, 258, 260, 265, 266, 267, 268, 269, 309, 313, 314, 315, 316, 317, 352, 356, 358, 360, 362, 367, 447, 448, 451, 458, 459, 462, 479, 480, 481, 482, 483, 485, 487, 492, 499, 505, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 540, 550, 551, 554, 558, 671, 672, 753, 755, 760, 784, 785, 787	L	
\EmFi@tile	225	\LaTeX	868
\EmFi@title	409, 414, 428, 547	\loop	809, 820
\EmFi@type	410, 419, 420, 421, 422, 423, 424, 425, 444, 455, 466, 523, 525, 527, 529, 531, 533, 535, 537	M	
\EmFi@view	357, 359, 361, 363, 779, 781	\MessageBreak	367, 487, 540, 541, 558, 746
\EmFi@visibletrue	411	N	
\empty	119, 356, 492, 498, 600, 601, 616, 629, 634, 677, 681, 720, 742, 772, 784	\NeedsTeXFormat	37, 98, 861
\end	94, 846, 869	\newif	239, 240, 241, 242, 243, 244, 245
\endcsname	115, 125, 146, 159, 162, 187, 211, 222, 246, 250, 261, 262, 263, 273, 301, 303, 310, 311, 321, 322, 327, 328, 332, 334, 404, 450, 461, 469, 475, 552, 555, 654, 655, 657, 661, 663, 665, 799, 845	\nopagenumbers	855
\endinput	134, 208, 218	P	
F		\PackageInfo	130
\futurelet	278, 297	\pdfcatalog	791
G		\pdfescapehex	680
\gdef	296, 390, 500	\pdfescapename	417, 496, 618, 631, 657, 660
		\pdfescapestring	674
		\pdffilemoddate	621
		\pdffilesize	592, 622
		\pdflastobj	612, 639, 648, 733, 792
		\pdfmdfivesum	623
		\pdfnames	732
		\pdfobj	607, 614, 626, 727, 770
		\pdfstrcmp	670, 685
		\pdfstringdef	40, 341, 344, 671
		\ProvidesPackage	99, 160

<b>R</b>		<b>\texttt</b> . . . . . 868
<b>\RangeCatcodeInvalid</b> . . . . .		<b>\the</b> 163, 169, 408, 612, 639, 648, 688,
. . . . . 818, 827, 828, 829, 830		695, 698, 701, 705, 733, 792, 812
<b>\repeat</b> . . . . . 816, 824		<b>\TMP@EnsureCode</b> . . . . . 166,
<b>\RequirePackage</b> . . . . . 101, 192, 864		173, 174, 175, 176, 177, 178,
<b>\reserved@a</b> . . . . . 276, 285		179, 180, 181, 182, 183, 184, 185
<b>\reserved@b</b> . . . . . 277, 287		<b>\toks@</b> 687, 688, 695, 698, 701, 705, 707
<b>\reserved@c</b> . . . . . 282, 285, 287, 290		<b>\tt</b> . . . . . 856
<b>\reserved@d</b> . . . . . 275, 284		
<b>\resetatcatcode</b> . . . . . 7	<b>U</b>	
<b>\RestoreCatcodes</b> . . 807, 810, 811, 842	<b>\usepackage</b> . . . . .	39, 42
<b>S</b>		<b>W</b>
<b>\SavedJobname</b> . . . . . 862, 865	<b>\write</b> . . . . .	127, 148
<b>\setkeys</b> . . . . . 384, 413, 573, 591	<b>X</b>	
<b>\space</b> . . . . . 378, 397,	<b>\x</b> . . . . .	115, 117, 119, 126,
504, 581, 639, 642, 648, 733, 792		130, 132, 147, 152, 159, 686, 693
<b>\stop</b> . . . . . 697, 704, 708	<b>Z</b>	
<b>T</b>		<b>\z@</b> . . . . . 247
<b>\Test</b> . . . . . 826, 844		
<b>\TeX</b> . . . . . 856		